

## Chapter 4 : IMPLEMENTATION

### 4.1 Model

#### Iterative design

Iterative design is a design methodology based on a cyclic process of prototyping, testing, analysing, and refining a product or process. Based on the results of testing the most recent iteration of a design, changes and refinements are made. This process is intended to ultimately improve the quality and functionality of a design. In iterative design, interaction with the designed system is used as a form of research for informing and evolving a project, as successive versions, or iterations of a design are implemented.

#### Iterative design process

The iterative design process may be applied throughout the new product development process. However, changes are easiest and less expensive to implement in the earliest stages of development. The first step in the iterative design process is to develop a prototype. The prototype should be evaluated by a focus group or a group not associated with the product in order to deliver non- biased opinions. Information from the focus group should be synthesized and incorporated into the next iteration of the design. The process should be repeated until user issues have been reduced to an acceptable level.

#### Application: Human computer interfaces

Iterative design is commonly used in the development of human computer interfaces. This allows designers to identify any usability issues that may arise in the user interface before it is put into wide use. Even the best usability experts cannot design perfect user interfaces in a single attempt, so a usability engineering lifecycle should be built around the concept of iteration. The typical steps of iterative design in user interfaces are as follows:

1. Complete an initial interface design
2. Present the design to several test users
3. Note any problems had by the test user
4. Refine interface to account for/fix the problems
5. Repeat steps 2-4 until user interface problems are resolved

Iterative design in user interfaces can be implemented in many ways. One common method of using iterative design in computer software is software testing. While this includes testing the product for functionality outside of the user interface, important feedback on the interface can be gained from subject testing early versions of a program. This allows software companies to release a better quality product to the public, and prevents the need

of product modification following its release. Iterative design in online (website) interfaces is a more continuous process, as website modification, after it has been released to the user, is far more viable than in software design. Often websites use their users as test subjects for interface design, making modifications based on recommendations from visitors to their sites.

## **Iterative design use**

Iterative design is a way of confronting the reality of unpredictable user needs and behaviours that can lead to sweeping and fundamental changes in a design. User testing will often show that even carefully evaluated ideas will be inadequate when confronted with a user test. Thus, it is important that the flexibility of the iterative design's implementation approach extends as far into the system as possible. Designers must further recognize that user testing results may suggest radical change that requires the designers to be prepared to completely abandon old ideas in favour of new ideas that are more equipped to suit user needs. Iterative design applies in many fields, from making knives to rockets. As an example consider the design of an electronic circuit that must perform a certain task, and must ultimately fit in a small space on a circuit board. It is useful to split these independent tasks into two smaller and simpler tasks, the functionality task, and the space and weight task. A breadboard is a useful way of implementing the electronic circuit on an interim basis, without having to worry about space and weight. Once the circuit works, improvements or incremental changes may be applied to the breadboard to increase or improve functionality over the original design. When the design is finalized, one can set about designing a proper circuit board meeting the space and weight criteria. Compacting the circuit on the circuit board requires that the wires and components be juggled around without changing their electrical characteristics. This juggling follows simpler rules than the design of the circuit itself, and is often automated. As far as possible off the shelf components are used, but where necessary for space or performance reasons, custom made components may be developed. Several instances of iterative design are as follows:

**Wiki** - A wiki is a natural repository for iterative design. The 'Page History' facility allows tracking back to prior versions. Modifications are mostly incremental, and leave substantial parts of the text unchanged.

**Common law** - The principle of legal precedent builds on past experience. This makes law a form of iterative design where there should be a clear audit trail of the development of legal thought.

**Evolution** - There is a parallel between iterative and the theory of Natural Selection. Both involve a trial and error process in which the most suitable design advances to the next generation, while less suitable designs perish by the wayside. Subsequent versions of a product should also get progressively better as its producers learn what works and what doesn't in a process of refinement and continuous improvement.

## Benefits

When properly applied, iterative design will ensure a product or process is the best solution possible. When applied early in the development stage, significant cost savings are possible. Other benefits to iterative design include:

1. Serious misunderstandings are made evident early in the lifecycle, when it's possible to react to them.
2. It enables and encourages user feedback, so as to elicit the system's real requirements.
3. Where the work is contracted, Iterative Design provides an incremental method for more effectively involving the client in the complexities that often surround the design process.
4. The development team is forced to focus on those issues that are most critical to the project, and team members are shielded from those issues that distract them from the project's real risks.
5. Continuous, iterative testing enables an objective assessment of the project's status.
6. Inconsistencies among requirements, designs, and implementations are detected early.
7. The workload of the team, especially the testing team, is spread out more evenly throughout the lifecycle.
8. This approach enables the team to leverage lessons learned, and therefore to continuously improve the process.
9. Stakeholders in the project can be given concrete evidence of the project's status throughout the lifecycle.

## 4.2 Data Flow Diagram

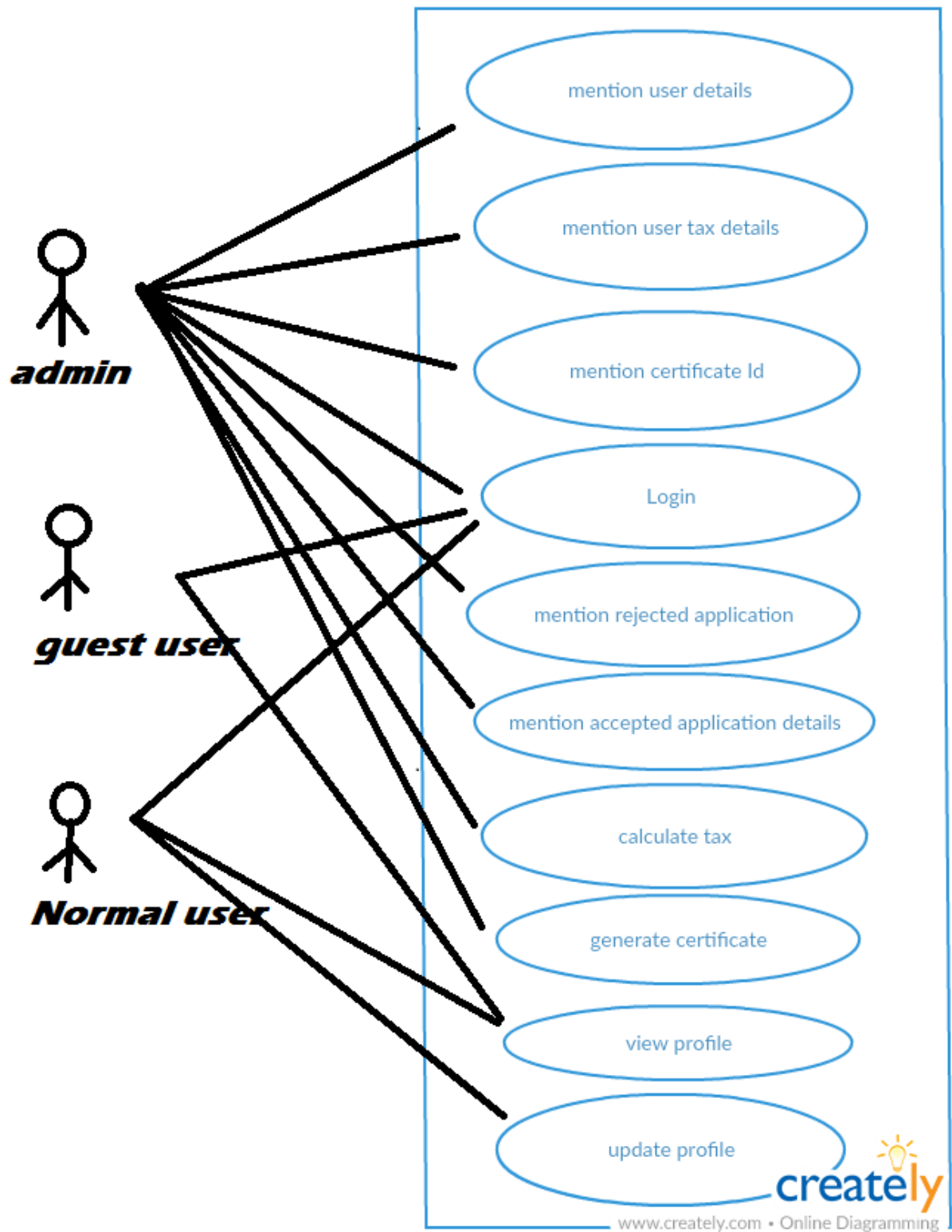
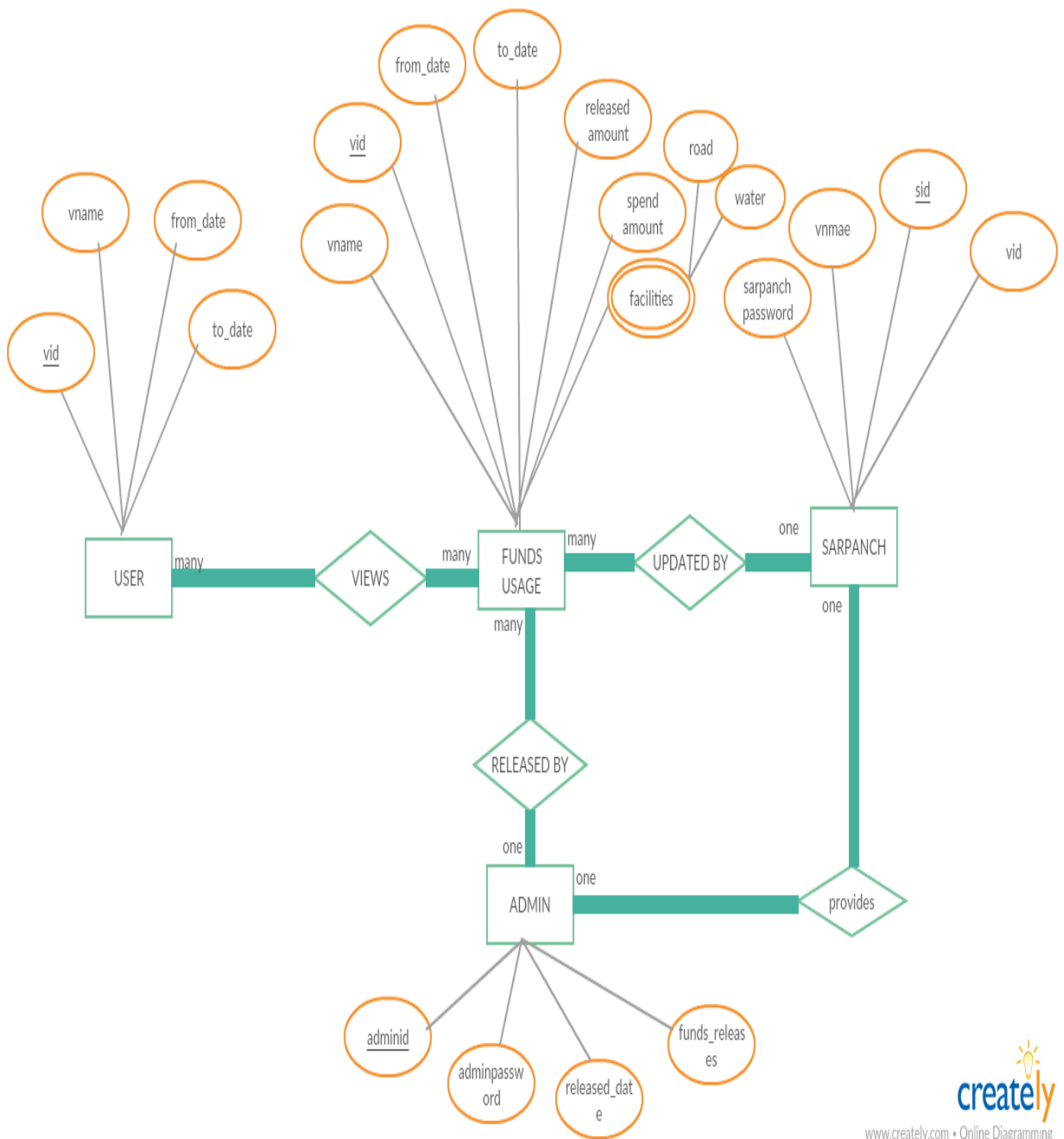
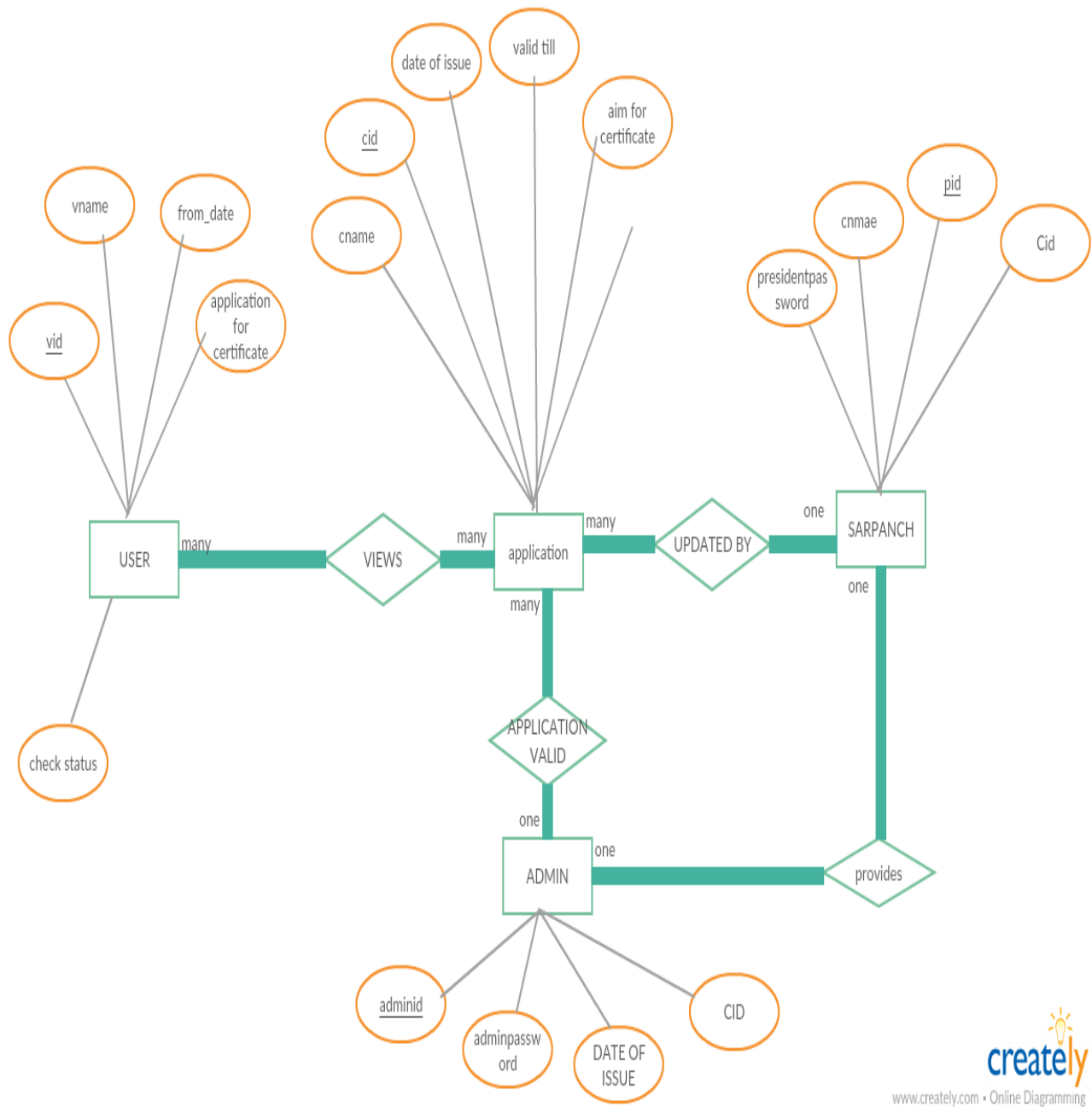


Figure 4.2.1: Use Case Diagram

### 4.3 ER Diagram



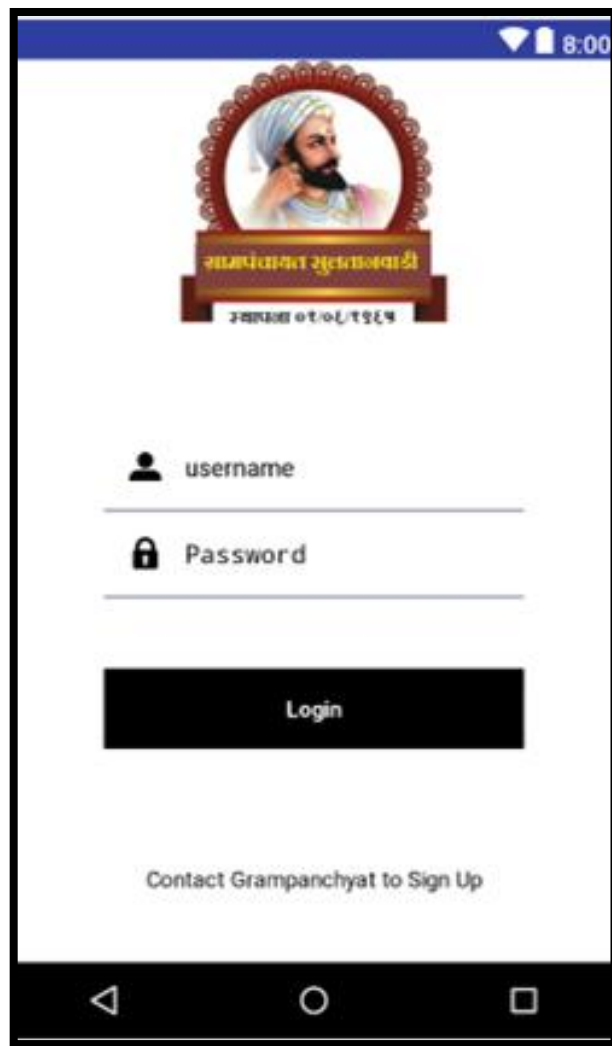
**Figure4.3.1: tax management system**



**Figure 4.3.2 application for certificate**

## 4.4 USER INTERFACE DESIGN

### LOGIN PAGE



Screenshot 5.3.1 screenshot of login page

### DESCRIPTION

This page allows user to log in in different type of user like admin or normal as per he registered previously by admin. This page contains abstract of the village some photo menu that includes different switchable webpage. This page has login form to get logged in.

### SOURCE CODE

```
package com.example.grampanchayat;

import android.annotation.SuppressLint;

import android.content.Context;
```

```
import android.content.Intent;

import android.nfc.Tag;

import android.support.annotation.NonNull;

import android.support.design.widget.TabLayout;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity {

    private static final String TAG = "LoginActivity";

    private EditText username;

    private EditText password1;

    private Button login1;

    private FirebaseAuth firebaseAuth;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_login);
```



```

firebaseAuth = FirebaseAuth.getInstance();

username = (EditText) findViewById(R.id.usernameetxt);

password1 = (EditText) findViewById(R.id.passwordtxt);

FirebaseUser user = firebaseAuth.getCurrentUser();

if (user != null) {

    finish();

    startActivity(new Intent(LoginActivity.this, UserHomeActivity.class));

}

login1 = (Button) findViewById(R.id.login);

login1.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View view) {

        if(username.getText().toString().isEmpty() )

        {

            Toast.makeText(getApplicationContext(), "वापरकर्तानाव आवश्यक", Toast.LENGTH_SHORT).show();

        }

        else if(password1.getText().toString().isEmpty())

        {

            Toast.makeText(getApplicationContext(), "पासवर्ड आवश्यक", Toast.LENGTH_SHORT).show();

        }

        else if(username.getText().toString().isEmpty() || password1.getText().toString().isEmpty())

        {

            Toast.makeText(getApplicationContext(), "वापरकर्तानाव आवश्यक || पासवर्ड आवश्यक", Toast.LENGTH_SHORT).show();

        }

        loginuser(username.getText().toString().trim(), password1.getText().toString().trim());

    }

}

```

```

}

);

}

private void loginuser(final String username, String password1) {

firebaseAuth.signInWithEmailAndPassword(username, password1)

.addOnCompleteListener(new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful() && username.equals("admin@gmail.com")) {

Toast.makeText(getApplicationContext(), "लॉगिन प्रशासक यशस्वी", Toast.LENGTH_SHORT).show();

startActivity(new Intent(LoginActivity.this, AdminHomeActivity.class));

} else {

Toast.makeText(getApplicationContext(), "लॉगिन वापरकर्ता यशस्वी", Toast.LENGTH_SHORT).show();

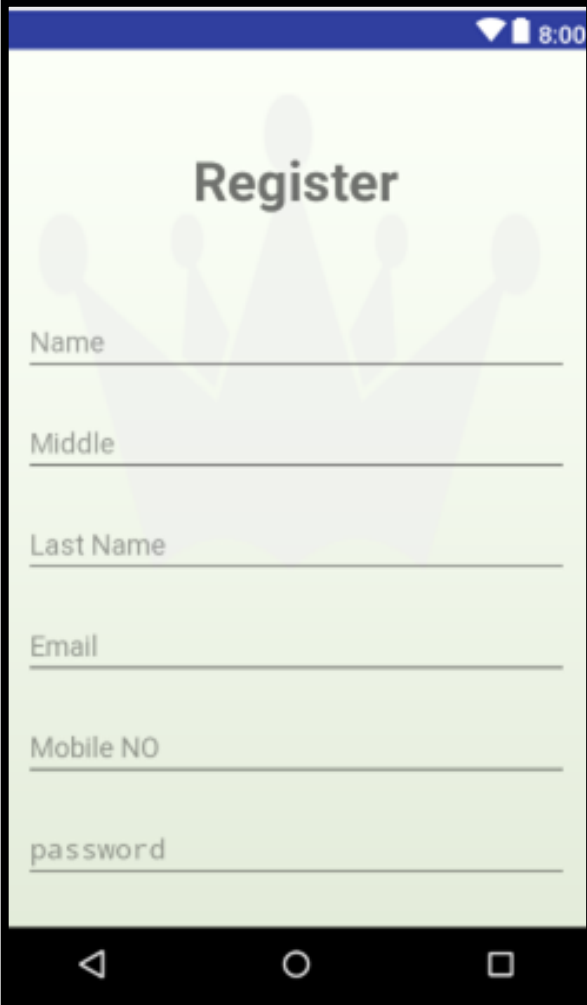
startActivity(new Intent(LoginActivity.this, RegisterActivity.class));

}

}

```

## REGISTER PAGE



Screenshot 5.3.2 register user

## DESCRIPTION

This page is for register new user. This page can only be accessed by admin. This page contains form to fill the detail of user to register. There are some validators are applied to form. And also menu is added to switch through the different pages.

## SOURCE CODE

```
package com.example.grampanchyat;

import android.app.ProgressDialog;
import android.content.Intent;
import android.content.IntentSender;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.net.Uri;
import android.provider.ContactsContract;
```

```

import android.provider.MediaStore;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.UserProfileChangeRequest;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;

import java.io.IOException;

public class RegisterActivity extends AppCompatActivity implements View.OnClickListener{

    EditText fname,mname,lname,email,mobile,password,hno,dob;
    Button submit;
    DatabaseReference databaseReference;
    FirebaseAuth mAuth;
    TextView textView;

    @Override

```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
    textView=(TextView)findViewById(R.id.imagetxt);
    SharedPreferences pre= getSharedPreferences("pre",MODE_PRIVATE);
    boolean firststart=pre.getBoolean("firststart",true);

    // inilazing firbase object
    mAuth= FirebaseAuth.getInstance();
    //if user not loged in
    if (mAuth.getCurrentUser()== null)
    {
        finish();
        startActivity(new Intent(this,LoginActivity.class));
    }
    databaseReference= FirebaseDatabase.getInstance().getReference("user");

    fname=(EditText)findViewById(R.id.fname);
    mname=(EditText)findViewById(R.id.mname);
    lname=(EditText)findViewById(R.id.lname);
    email=(EditText)findViewById(R.id.email);
    mobile=(EditText)findViewById(R.id.mobile);
    password=(EditText)findViewById(R.id.password);
    hno=(EditText)findViewById(R.id.hno);
    dob=(EditText)findViewById(R.id.dob);
    submit=(Button)findViewById(R.id.submit);
    FirebaseUser user=mAuth.getCurrentUser();

    submit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            adduser();
        }
    });
}

private void adduser()
{

```

```

String First=fname.getText().toString().trim();
String Middle=mname.getText().toString().trim();
String Last=lname.getText().toString().trim();
String Email=email.getText().toString().trim();
String Mobile=mobile.getText().toString().trim();
String Password=password.getText().toString().trim();
String Hno=hno.getText().toString().trim();
String Dob=dob.getText().toString().trim();

if (!TextUtils.isEmpty(First))
{

    user user1=new user(First,Middle,Last,Email,Mobile>Password,Hno,Dob);
    FirebaseAuth user=mAuth.getCurrentUser();

    databaseReference.child(user.getId()).setValue(user1);
    Toast.makeText(this,"information saved",Toast.LENGTH_LONG).show() ;
    startActivity(new Intent(RegisterActivity.this, User_home_activity.class));
    SharedPreferences pre=getSharedPreferences("pre",MODE_PRIVATE);
    SharedPreferences.Editor editor=pre.edit();
    editor.putBoolean("firststart",false);
    editor.apply();
}
else { Toast.makeText(this,"enter name",Toast.LENGTH_SHORT).show();
}

/* if (!TextUtils.isEmpty(Middle)
{
}
else { Toast.makeText(this,"enter Middle",Toast.LENGTH_SHORT).show();}
if (!TextUtils.isEmpty(Last))
{
}
else { Toast.makeText(this,"enter last",Toast.LENGTH_SHORT).show();}
if (!TextUtils.isEmpty(Email))
{
}
else { Toast.makeText(this,"enter email",Toast.LENGTH_SHORT).show();}
if (!TextUtils.isEmpty(Mobile))
{
}
else { Toast.makeText(this,"enter mobile",Toast.LENGTH_SHORT).show();}
if (!TextUtils.isEmpty>Password))

```

```
    {}  
    else {Toast.makeText(this,"enter password",Toast.LENGTH_SHORT).show();}  
    if (!TextUtils.isEmpty(Hno))  
    {  
    {}  
    else {Toast.makeText(this,"enter hno",Toast.LENGTH_SHORT).show();}  
    if (!TextUtils.isEmpty(Dob))  
    {  
    {}  
    else {Toast.makeText(this,"enter dob",Toast.LENGTH_SHORT).show();}  
  
    */  
  
    }  
  
    @Override  
    public void onClick(View view) {  
  
    }  
  
    }
```